COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco
University of Massachusetts Amherst. Fall 2019.
Lecture 18
• Problem Set 3 on Spectral Methods due **this Friday at 8pm**.
• Can turn in without penalty until Sunday at 11:59pm.
Last Class:

- Power method for computing the top singular vector of a matrix.
- High level discussion of Krylov methods, block versions for computing more singular vectors.
Last Class:

- Power method for computing the top singular vector of a matrix.
- High level discussion of Krylov methods, block versions for computing more singular vectors.
- Power method is an iterative algorithm for solving the non-convex optimization problem:
  \[ \max_{\vec{v}} \quad \vec{v}^T X^T X \vec{v}, \]
  \[ \vec{v} : \|\vec{v}\|_2^2 \leq 1 \]

This Class (and until Thanksgiving):

- More general iterative algorithms for optimization, specifically gradient descent and its variants.
- What are they methods, when are they applied, and how do you analyze there performance?
- Small taste of what you can find in COMPSCI 590OP or 690OP.
Discrete (Combinatorial) Optimization: (traditional CS algorithms)

- Graph Problems: min-cut, max flow, shortest path, matchings, maximum independent set, traveling salesman problem
- Problems with discrete constraints or outputs: bin-packing, scheduling, sequence alignment, submodular maximization
- Generally searching over a finite but exponentially large set of possible solutions. Many of these problems are NP-Hard.
Discrete (Combinatorial) Optimization: (traditional CS algorithms)

- Graph Problems: min-cut, max flow, shortest path, matchings, maximum independent set, traveling salesman problem
- Problems with discrete constraints or outputs: bin-packing, scheduling, sequence alignment, submodular maximization
- Generally searching over a finite but exponentially large set of possible solutions. Many of these problems are NP-Hard.
Discrete (Combinatorial) Optimization: (traditional CS algorithms)

- Graph Problems: min-cut, max flow, shortest path, matchings, maximum independent set, traveling salesman problem
- Problems with discrete constraints or outputs: bin-packing, scheduling, sequence alignment, submodular maximization
- Generally searching over a finite but exponentially large set of possible solutions. Many of these problems are NP-Hard.

Continuous Optimization: (not covered in core CS curriculum. Touched on in ML/advanced algorithms, maybe.)

- Unconstrained convex and non-convex optimization.
- Linear programming, quadratic programming, semidefinite programming
CONTINUOUS OPTIMIZATION EXAMPLES

\[ f(\theta) \]

\[ \theta \in \mathbb{R} \]

\[ \theta^* \]
CONTINUOUS OPTIMIZATION EXAMPLES

\[ f(\theta) \quad \theta \in \mathbb{R} \]

\[ f(\theta) \quad \theta \in \mathbb{R}^2 \]

\[ f(\theta) \quad \theta \in \mathbb{R}^2 \]
Given some function $f : \mathbb{R}^d \to \mathbb{R}$, find $\vec{\theta}_*$ with:

$$f(\vec{\theta}_*) = \min_{\vec{\theta} \in \mathbb{R}^d} f(\vec{\theta})$$
Given some function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, find $\vec{\theta}_\star$ with:

$$f(\vec{\theta}_\star) = \min_{\vec{\theta} \in \mathbb{R}^d} f(\vec{\theta}) + \epsilon$$

Typically up to some small approximation factor.
Given some function $f: \mathbb{R}^d \to \mathbb{R}$, find $\vec{\theta}_\star$ with:

$$f(\vec{\theta}_\star) = \min_{\vec{\theta} \in \mathbb{R}^d} f(\vec{\theta}) + \epsilon$$

Typically up to some small approximation factor.

Often under some constraints:

- $\|\vec{\theta}\|_2 \leq 1$, $\|\vec{\theta}\|_1 \leq 1$.
- $A\vec{\theta} \leq \vec{b}$, $\vec{\theta}^T A\vec{\theta} \geq 0$.
- $\vec{1}^T \vec{\theta} = \sum_{i=1}^{d} \vec{\theta}(i) \leq c$. 
Modern machine learning centers around continuous optimization. Typical Set Up (supervised machine learning):

- Have a model, which is a function mapping inputs to predictions (neural network, linear function, low-degree polynomial etc).
- The model is parameterized by a parameter vector (weights in a neural network, coefficients in a linear function or polynomial).
- Want to train this model on input data, by picking a parameter vector such that the model does a good job mapping inputs to predictions on your training data.

This training step is typically formulated as a continuous optimization problem.
Modern machine learning centers around continuous optimization.
Modern machine learning centers around continuous optimization.

**Typical Set Up: (supervised machine learning)**

- Have a **model**, which is a function mapping inputs to predictions (neural network, linear function, low-degree polynomial etc).
- The model is parameterized by a **parameter vector** (weights in a neural network, coefficients in a linear function or polynomial)
- Want to **train** this model on input data, by picking a parameter vector such that the model does a good job mapping inputs to predictions on your training data.
Modern machine learning centers around continuous optimization.

**Typical Set Up: (supervised machine learning)**

- Have a **model**, which is a function mapping inputs to predictions (neural network, linear function, low-degree polynomial etc).
- The model is parameterized by a **parameter vector** (weights in a neural network, coefficients in a linear function or polynomial).
- Want to **train** this model on input data, by picking a parameter vector such that the model does a good job mapping inputs to predictions on your training data.

This training step is typically formulated as a continuous optimization problem.
Example 1: Linear Regression
Example 1: Linear Regression

Model: $M_{\vec{\theta}} : \mathbb{R}^d \rightarrow \mathbb{R}$ with $M_{\vec{\theta}}(\vec{x}) \overset{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle$
Example 1: Linear Regression

Model: $M_{\theta} : \mathbb{R}^d \rightarrow \mathbb{R}$ with $M_{\theta}(\vec{x}) \overset{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle = \vec{\theta}(1) \cdot \vec{x}(1) + \ldots + \vec{\theta}(d) \cdot \vec{x}(d)$. 
Example 1: Linear Regression

Model: $M_{\vec{\theta}} : \mathbb{R}^d \rightarrow \mathbb{R}$ with $M_{\vec{\theta}}(\vec{x}) \overset{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle = \vec{\theta}(1) \cdot x(1) + \ldots + \vec{\theta}(d) \cdot x(d)$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}^d$ (the regression coefficients)
Example 1: Linear Regression

Model: \( M_{\theta} : \mathbb{R}^d \rightarrow \mathbb{R} \) with \( M_{\theta}(\vec{x}) \overset{\text{def}}{=} \langle \theta, \vec{x} \rangle = \theta(1) \cdot x(1) + \ldots + \theta(d) \cdot x(d) \).

Parameter Vector: \( \vec{\theta} \in \mathbb{R}^d \) (the regression coefficients)

Optimization Problem: Given data points (training points) \( \vec{x}_1, \ldots, \vec{x}_n \) (the rows of data matrix \( X \in \mathbb{R}^{n \times d} \)) and labels \( y_1, \ldots, y_n \in \mathbb{R} \), find \( \vec{\theta}_* \) minimizing the loss function:

\[
L(\vec{\theta}, X) = \sum_{i=1}^{n} \ell(M_{\theta}(\vec{x}_i), y_i)
\]

where \( \ell \) is some measurement of how far \( M_{\theta}(\vec{x}_i) \) is from \( y_i \).
Example 1: Linear Regression

Model: $M_{\vec{\theta}} : \mathbb{R}^d \rightarrow \mathbb{R}$ with $M_{\vec{\theta}}(\vec{x}) \overset{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle = \theta(1) \cdot x(1) + \ldots + \theta(d) \cdot x(d)$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}^d$ (the regression coefficients)

Optimization Problem: Given data points (training points) $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $X \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n \in \mathbb{R}$, find $\vec{\theta}_*$ minimizing the loss function:

$$L(\vec{\theta}, X) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

where $\ell$ is some measurement of how far $M_{\vec{\theta}}(\vec{x}_i)$ is from $y_i$.

- $\ell(M_{\vec{\theta}}(\vec{x}_i), y_i) = (M_{\vec{\theta}}(\vec{x}_i) - y_i)^2$ (least squares regression)
- $y_i \in \{-1, 1\}$ and $\ell(M_{\vec{\theta}}(\vec{x}_i), y_i) = \ln (1 + \exp(-y_i M_{\vec{\theta}}(\vec{x}_i)))$ (logistic regression)
Optimization in ML

Example 1: Linear Regression

Model: \( M_{\vec{\theta}} : \mathbb{R}^d \rightarrow \mathbb{R} \) with \( M_{\vec{\theta}}(\vec{x}) \overset{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle = \vec{\theta}(1) \cdot x(1) + \ldots + \vec{\theta}(d) \cdot x(d) \).

Parameter Vector: \( \vec{\theta} \in \mathbb{R}^d \) (the regression coefficients)

Optimization Problem: Given data points (training points) \( \vec{x}_1, \ldots, \vec{x}_n \) (the rows of data matrix \( X \in \mathbb{R}^{n \times d} \)) and labels \( y_1, \ldots, y_n \in \mathbb{R} \), find \( \vec{\theta}_* \) minimizing the loss function:

\[
L(\vec{\theta}, X) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i) + R(\vec{\theta})
\]

where \( \ell \) is some measurement of how far \( M_{\vec{\theta}}(\vec{x}_i) \) is from \( y_i \).

- \( \ell(M_{\vec{\theta}}(\vec{x}_i), y_i) = (M_{\vec{\theta}}(\vec{x}_i) - y_i)^2 \) (least squares regression)
- \( y_i \in \{-1, 1\} \) and \( \ell(M_{\vec{\theta}}(\vec{x}_i), y_i) = \ln(1 + \exp(-y_i M_{\vec{\theta}}(\vec{x}_i))) \) (logistic regression)
Example 1: Linear Regression

Model: \( M_\theta : \mathbb{R}^d \rightarrow \mathbb{R} \) with \( M_\theta(\vec{x}) \overset{\text{def}}{=} \langle \theta, \vec{x} \rangle = \theta(1) \cdot x(1) + \ldots + \theta(d) \cdot x(d) \).

Parameter Vector: \( \vec{\theta} \in \mathbb{R}^d \) (the regression coefficients)

Optimization Problem: Given data points (training points) \( \vec{x}_1, \ldots, \vec{x}_n \) (the rows of data matrix \( \mathbf{X} \in \mathbb{R}^{n \times d} \)) and labels \( y_1, \ldots, y_n \in \mathbb{R} \), find \( \vec{\theta}_* \) minimizing the loss function:

\[
L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \ell(M_\theta(\vec{x}_i), y_i) + \lambda \| \vec{\theta} \|_2^2
\]

where \( \ell \) is some measurement of how far \( M_\theta(\vec{x}_i) \) is from \( y_i \).

- \( \ell(M_\theta(\vec{x}_i), y_i) = (M_\theta(\vec{x}_i) - y_i)^2 \) (least squares regression)
- \( y_i \in \{-1, 1\} \) and \( \ell(M_\theta(\vec{x}_i), y_i) = \ln (1 + \exp(-y_i M_\theta(\vec{x}_i))) \) (logistic regression)
Example 2: Neural Networks

Model: \( \mathbf{M} : \mathbb{R}^d \rightarrow \mathbb{R} \). \( \mathbf{M}(\mathbf{x}) = \langle \mathbf{w}_{\text{out}}, (W_2(W_1(\mathbf{x}))) \rangle \).

Parameter Vector: \( \mathbf{w}_{\text{out}} \) (the weights on every edge)

Optimization Problem: Given data points \( \mathbf{x}_1, \ldots, \mathbf{x}_n \) and labels \( y_1, \ldots, y_n \in \mathbb{R} \), find \( \mathbf{w}_{\text{out}} \) minimizing the loss function:

\[
L(\mathbf{w}; X) = \sum_{i=1}^{n} \ell(M(\mathbf{x}_i); y_i)
\]
Example 2: Neural Networks

Model: \( M_{\theta} : \mathbb{R}^d \rightarrow \mathbb{R} \).

Parameter Vector: \( \tilde{\theta} \in \mathbb{R}^{(# \text{ edges})} \) (the weights on every edge)
Example 2: Neural Networks

Model: $M_{\tilde{\theta}} : \mathbb{R}^d \rightarrow \mathbb{R}$. $M_{\tilde{\theta}}(\tilde{x}) = \langle \tilde{W}_{out}, \sigma(W_2 \sigma(W_1 \tilde{x})) \rangle$.

Parameter Vector: $\tilde{\theta} \in \mathbb{R}^{(# \text{ edges})}$ (the weights on every edge)
Example 2: Neural Networks

Model: $M_{\vec{\theta}} : \mathbb{R}^d \rightarrow \mathbb{R}$. $M_{\vec{\theta}}(\vec{x}) = \langle \vec{w}_{out}, \sigma(\vec{W}_2 \sigma(\vec{W}_1 \vec{x})) \rangle$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}(\# \text{ edges})$ (the weights on every edge)

Optimization Problem: Given data points $\vec{x}_1, \ldots, \vec{x}_n$ and labels $y_1, \ldots, y_n \in \mathbb{R}$, find $\vec{\theta}_*$ minimizing the loss function:

$$L(\vec{\theta}, X) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$
$L(\vec{\theta}, X) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$

- **Supervised** means we have labels $y_1, \ldots, y_n$ for the training points.
\[ L(\theta, X) = \sum_{i=1}^{n} \ell(M_{\theta}(\tilde{x}_i), y_i) \]

- **Supervised** means we have labels \( y_1, \ldots, y_n \) for the training points.
- Solving the final optimization problem has many different names: likelihood maximization, empirical risk minimization, minimizing training loss, etc.
Optimization in ML

\[ L(\vec{\theta}, X) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i) \]

- **Supervised** means we have labels \( y_1, \ldots, y_n \) for the training points.
- Solving the final optimization problem has many different names: likelihood maximization, empirical risk minimization, minimizing training loss, etc.
- Continuous optimization is also very common in unsupervised learning.
Optimization in ML

\[ L(\tilde{\theta}, X) = \sum_{i=1}^{n} \ell(M_{\tilde{\theta}}(\tilde{x}_i), y_i) \]

- **Supervised** means we have labels \( y_1, \ldots, y_n \) for the training points.
- Solving the final optimization problem has many different names: likelihood maximization, empirical risk minimization, minimizing training loss, etc.
- Continuous optimization is also very common in unsupervised learning. (PCA, spectral clustering, etc.)
\[ L(\vec{\theta}, X) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i) \]

- **Supervised** means we have labels \( y_1, \ldots, y_n \) for the training points.

- Solving the final optimization problem has many different names: likelihood maximization, empirical risk minimization, minimizing training loss, etc.

- Continuous optimization is also very common in unsupervised learning. (PCA, spectral clustering, etc.)

- **Generalization** tries to explain why minimizing the loss \( L(\vec{\theta}, X) \) on the *training points* minimizes the loss on future *test points*. I.e., makes us have good predictions on future inputs.
Choice of optimization algorithm for minimizing $f(\vec{\theta})$ will depend on many things:

- The form of $f$ (in ML, depends on the model & loss function).
- Any constraints on $\vec{\theta}$ (e.g., $\|\vec{\theta}\| < c$).
- Other constraints, such as memory constraints.
Choice of optimization algorithm for minimizing $f(\vec{\theta})$ will depend on many things:

- The form of $f$ (in ML, depends on the model & loss function).
- Any constraints on $\vec{\theta}$ (e.g., $\|\vec{\theta}\| < c$).
- Other constraints, such as memory constraints.

$$L(\vec{\theta}, X) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

What are some popular optimization algorithms?
This class: Gradient descent (and some important variants)
This class: Gradient descent (and some important variants)

• An extremely simple greedy iterative method, that can be applied to almost any continuous function we care about optimizing.
This class: Gradient descent (and some important variants)

- An extremely simple greedy iterative method, that can be applied to almost any continuous function we care about optimizing.
- Often not the ‘best’ choice for any given function, but it is the approach of choice in ML since it is simple, general, and often works very well.
**This class**: Gradient descent (and some important variants)

- An extremely simple greedy iterative method, that can be applied to almost any continuous function we care about optimizing.
- Often not the ‘best’ choice for any given function, but it is the **approach** of choice in ML since it is simple, general, and often works very well.
- At each step, tries to move towards the lowest nearby point in the function that is can – in the direction of the gradient.
This class: Gradient descent (and some important variants)

• An extremely simple greedy iterative method, that can be applied to almost any continuous function we care about optimizing.

• Often not the ‘best’ choice for any given function, but it is the approach of choice in ML since it is simple, general, and often works very well.

• At each step, tries to move towards the lowest nearby point in the function that is can – in the direction of the gradient.
Let $\vec{e}_i \in \mathbb{R}^d$ denote the $i^{th}$ standard basis vector, $\vec{e}_i = [0, 0, 1, 0, 0, \ldots, 0]$. 1 at position $i$. 
Let $\vec{e}_i \in \mathbb{R}^d$ denote the $i^{th}$ standard basis vector, $\vec{e}_i = [0, 0, 1, 0, 0, \ldots, 0]$.  

1 at position $i$

Partial Derivative:

$$\frac{\partial f}{\partial \theta(i)} = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon \cdot \vec{e}_i) - f(\vec{\theta})}{\epsilon}.$$
Let $\vec{e}_i \in \mathbb{R}^d$ denote the $i^{th}$ standard basis vector,

$$\vec{e}_i = [0, 0, 1, 0, 0, \ldots, 0].$$

1 at position $i$

Partial Derivative:

$$\frac{\partial f}{\partial \theta(i)} = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon \cdot \vec{e}_i) - f(\vec{\theta})}{\epsilon}. $$

Directional Derivative:

$$D_{\vec{v}} f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon \vec{v}) - f(\vec{\theta})}{\epsilon}. $$
Gradient: Just a ‘list’ of the partial derivatives.

\[ \vec{\nabla} f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \theta(1)} \\ \frac{\partial f}{\partial \theta(2)} \\ \vdots \\ \frac{\partial f}{\partial \theta(d)} \end{bmatrix} \]
Gradient: Just a ‘list’ of the partial derivatives.

\[ \nabla f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \theta(1)} \\
\frac{\partial f}{\partial \theta(2)} \\
\vdots \\
\frac{\partial f}{\partial \theta(d)} \end{bmatrix} \]

Directional Derivative in Terms of the Gradient:

\[ D_{\vec{v}} f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon \vec{v}) - f(\vec{\theta})}{\epsilon} \]
**Multivariate Calculus Review**

**Gradient:** Just a ‘list’ of the partial derivatives.

\[ \vec{\nabla} f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \theta(1)} \\ \frac{\partial f}{\partial \theta(2)} \\ \vdots \\ \frac{\partial f}{\partial \theta(d)} \end{bmatrix} \]

**Directional Derivative in Terms of the Gradient:**

\[ D_{\vec{v}} f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon (\vec{e}_1 \cdot \vec{v}(1) + \vec{e}_2 \cdot \vec{v}(2) + \ldots + \vec{e}_d \cdot \vec{v}(d))) - f(\vec{\theta})}{\epsilon} \]
Gradient: Just a ‘list’ of the partial derivatives.

\[ \vec{\nabla} f(\vec{\theta}) = \left[ \begin{array}{c} \frac{\partial f}{\partial \theta(1)} \\ \frac{\partial f}{\partial \theta(2)} \\ \vdots \\ \frac{\partial f}{\partial \theta(d)} \end{array} \right] \]

Directional Derivative in Terms of the Gradient:

\[ D_{\vec{v}} f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon (\vec{e}_1 \cdot \vec{v}(1) + \vec{e}_2 \cdot \vec{v}(2) + \ldots + \vec{e}_d \cdot \vec{v}(d)) - f(\vec{\theta})}{\epsilon} \]
Gradient: Just a ‘list’ of the partial derivatives.

$$\vec{\nabla} f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \theta(1)} \\ \frac{\partial f}{\partial \theta(2)} \\ \vdots \\ \frac{\partial f}{\partial \theta(d)} \end{bmatrix}$$

Directional Derivative in Terms of the Gradient:

$$D_{\vec{v}} f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon (\vec{e}_1 \cdot \vec{v}(1) + \vec{e}_2 \cdot \vec{v}(2) + \ldots + \vec{e}_d \cdot \vec{v}(d))) - f(\vec{\theta})}{\epsilon}$$

$$\approx \vec{v}(1) \cdot \frac{\partial f}{\partial \theta(1)}$$
Gradient: Just a ‘list’ of the partial derivatives.

\[ \vec{\nabla} f(\vec{\theta}) = \left[ \begin{array}{c} \frac{\partial f}{\partial \theta(1)} \\
\frac{\partial f}{\partial \theta(2)} \\
\vdots \\
\frac{\partial f}{\partial \theta(d)} \end{array} \right] \]

Directional Derivative in Terms of the Gradient:

\[ D_{\vec{v}} f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon (\vec{e}_1 \cdot \vec{v}(1) + \vec{e}_2 \cdot \vec{v}(2) + \ldots + \vec{e}_d \cdot \vec{v}(d))) - f(\vec{\theta})}{\epsilon} \]

\[ \approx \vec{v}(1) \cdot \frac{\partial f}{\partial \theta(1)} + \vec{v}(2) \cdot \frac{\partial f}{\partial \theta(2)} + \ldots + \vec{v}(d) \cdot \frac{\partial f}{\partial \theta(d)} \]
**Gradient:** Just a ‘list’ of the partial derivatives.

\[
\vec{\nabla} f(\vec{\theta}) = \left[ \begin{array}{c} \frac{\partial f}{\partial \theta(1)} \\ \frac{\partial f}{\partial \theta(2)} \\ \vdots \\ \frac{\partial f}{\partial \theta(d)} \end{array} \right]
\]

**Directional Derivative in Terms of the Gradient:**

\[
D_{\vec{v}} f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon (\vec{e}_1 \cdot \vec{v}(1) + \vec{e}_2 \cdot \vec{v}(2) + \cdots + \vec{e}_d \cdot \vec{v}(d))) - f(\vec{\theta})}{\epsilon}
\]

\[
\approx \vec{v}(1) \cdot \frac{\partial f}{\partial \theta(1)} + \vec{v}(2) \cdot \frac{\partial f}{\partial \theta(2)} + \cdots + \vec{v}(d) \cdot \frac{\partial f}{\partial \theta(d)}
\]

\[
= \langle \vec{v}, \vec{\nabla} f(\vec{\theta}) \rangle.
\]
Often the functions we are trying to optimize are very complex (e.g., a neural network). We will assume access to:

**Function Evaluation**: Can compute $f(\theta)$ for any $\theta$.

**Gradient Evaluation**: Can compute $\nabla f(\theta)$ for any $\theta$. 
Often the functions we are trying to optimize are very complex (e.g., a neural network). We will assume access to:

**Function Evaluation:** Can compute $f(\theta)$ for any $\theta$.

**Gradient Evaluation:** Can compute $\nabla f(\theta)$ for any $\theta$.

In neural networks:

- Function evaluation is called a **forward pass** (propogate an input through the network).
- Gradient evaluation is called a **backward pass** (compute the gradient via chain rule, using backpropagation).
Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $X \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$), find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, X) = \sum_{i=1}^{n} \left( \vec{\theta}^T \vec{x}_i - y_i \right)^2$$
Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $X \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$), find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, X) = \sum_{i=1}^{n} \left( \vec{\theta}^T \vec{x}_i - y_i \right)^2 = \|X\vec{\theta} - \vec{y}\|_2^2.$$ 

By Chain rule:

$$\frac{\partial L(\vec{\theta}, X)}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left( \vec{\theta}^T \vec{x}_i - y_i \right) \cdot \frac{\partial \left( \vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)}$$
Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$), find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \left( \vec{\theta}^T \vec{x}_i - y_i \right)^2 = \| \mathbf{X} \vec{\theta} - \vec{y} \|_2^2.$$ 

By Chain rule:

$$\frac{\partial L(\vec{\theta}, \mathbf{X})}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left( \vec{\theta}^T \vec{x}_i - y_i \right) \cdot \frac{\partial \left( \vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)}$$

$$\frac{\partial \left( \vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)} = \frac{\partial (\vec{\theta}^T \vec{x}_i)}{\partial \vec{\theta}(j)}$$
Running Example: Least squares regression.

Given input points $\bar{x}_1, \ldots, \bar{x}_n$ (the rows of data matrix $X \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\bar{y} \in \mathbb{R}^n$), find $\bar{\theta}_*$ minimizing:

$$L(\bar{\theta}, X) = \sum_{i=1}^{n} \left( \bar{\theta}^T \bar{x}_i - y_i \right)^2 = \|X \bar{\theta} - \bar{y}\|_2^2.$$

By Chain rule:

$$\frac{\partial L(\bar{\theta}, X)}{\partial \bar{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left( \bar{\theta}^T \bar{x}_i - y_i \right) \cdot \frac{\partial \left( \bar{\theta}^T \bar{x}_i - y_i \right)}{\partial \bar{\theta}(j)}$$

$$\frac{\partial \left( \bar{\theta}^T \bar{x}_i - y_i \right)}{\partial \bar{\theta}(j)} = \frac{\partial (\theta^T \bar{x}_i)}{\partial \bar{\theta}(j)} = \lim_{\epsilon \to 0} \frac{\theta^T \bar{x}_i - (\theta + \epsilon \bar{e}_j)^T \bar{x}_i}{\epsilon}$$
Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $X \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$), find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, X) = \sum_{i=1}^{n} \left( \vec{\theta}^T \vec{x}_i - y_i \right)^2 = \| X \vec{\theta} - \vec{y} \|_2^2.$$

By Chain rule:

$$\frac{\partial L(\vec{\theta}, X)}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left( \vec{\theta}^T \vec{x}_i - y_i \right) \cdot \frac{\partial \left( \vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)}$$

$$\frac{\partial \left( \vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)} = \frac{\partial (\vec{\theta}^T \vec{x}_i)}{\partial \vec{\theta}(j)} = \lim_{\epsilon \to 0} \frac{\theta^T \vec{x}_i - (\theta + \epsilon \vec{e}_j)^T \vec{x}_i}{\epsilon} = \lim_{\epsilon \to 0} \frac{\epsilon \vec{e}_j^T \vec{x}_i}{\epsilon}$$
Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $X \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$), find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, X) = \sum_{i=1}^{n} \left( \vec{\theta}^T \vec{x}_i - y_i \right)^2 = \|X\vec{\theta} - \vec{y}\|_2^2.$$ 

By Chain rule:

$$\frac{\partial L(\vec{\theta}, X)}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left( \vec{\theta}^T \vec{x}_i - y_i \right) \cdot \frac{\partial \left( \vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)}$$

$$\frac{\partial \left( \vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)} = \frac{\partial (\theta^T \vec{x}_i)}{\partial \vec{\theta}(j)} = \lim_{\epsilon \to 0} \frac{\theta^T \vec{x}_i - (\theta + \epsilon \vec{e}_j)^T \vec{x}_i}{\epsilon} = \lim_{\epsilon \to 0} \frac{\epsilon \vec{e}_j^T \vec{x}_i}{\epsilon} = \vec{x}_i(j).$$
**Running Example:** Least squares regression.

Given input points $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $X \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$), find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, X) = \sum_{i=1}^{n} \left( \vec{\theta}^T \vec{x}_i - y_i \right)^2 = \|X\vec{\theta} - \vec{y}\|_2^2.$$ 

By Chain rule:

$$\frac{\partial L(\vec{\theta}, X)}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left( \vec{\theta}^T \vec{x}_i - y_i \right) \cdot \frac{\partial \left( \vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)}$$

$$= \sum_{i=1}^{n} 2 \cdot \left( \vec{\theta}^T \vec{x}_i - y_i \right) \vec{x}_i(j)$$

$$\frac{\partial \left( \vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)} = \frac{\partial (\vec{\theta}^T \vec{x}_i)}{\partial \vec{\theta}(j)} = \lim_{\epsilon \to 0} \frac{\theta^T \vec{x}_i - (\theta + \epsilon \vec{e}_j)^T \vec{x}_i}{\epsilon} = \lim_{\epsilon \to 0} \frac{\epsilon \vec{e}_j^T \vec{x}_i}{\epsilon} = \vec{x}_i(j).$$
Partial derivative for least squares regression:

\[
\frac{\partial L(\vec{\theta}, X)}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot (\vec{\theta}^T \vec{x}_i - y_i) \vec{x}_i(j).
\]
Partial derivative for least squares regression:

$$\frac{\partial L(\vec{\theta}, X)}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot (\vec{\theta}^T \vec{x}_i - y_i) \vec{x}_i(j).$$

$$\nabla L(\vec{\theta}, X) = \sum_{i=1}^{n} 2 \cdot (\vec{\theta}^T \vec{x}_i - y_i) \vec{x}_i$$
Partial derivative for least squares regression:

\[
\frac{\partial L(\theta, X)}{\partial \theta(j)} = \sum_{i=1}^{n} 2 \cdot \left( \theta^T \bar{x}_i - y_i \right) \bar{x}_i(j).
\]

\[
\nabla L(\theta, X) = \sum_{i=1}^{n} 2 \cdot \left( \theta^T \bar{x}_i - y_i \right) \bar{x}_i
\]

\[
= X^T (X\theta - \bar{y}).
\]
Gradient for least squares regression via linear algebraic approach:

$$\nabla L(\vec{\theta}, \vec{X}) = \nabla \| \vec{X}\vec{\theta} - \vec{y} \|_2^2$$
Gradient descent is a greedy iterative optimization algorithm: Starting at $\theta^{(0)}$, in each iteration let $\theta^{(i)} = \theta^{(i-1)} + \eta \nabla f$, where $\eta$ is a (small) ‘step size’ and $\nabla f$ is a direction chosen to minimize $f(\theta^{(i-1)} + \eta \nabla f)$. 

$$D_{\nabla f}(\theta) = \lim_{\epsilon \to 0} \frac{f(\theta + \epsilon \nabla f) - f(\theta)}{\epsilon} :$$

We want to choose $\nabla f$ minimizing $\langle \nabla f; \theta \rangle$ – i.e., pointing in the direction of $\nabla f$ but with the opposite sign.
Gradient descent is a **greedy** iterative optimization algorithm: Starting at $\vec{\theta}^{(0)}$, in each iteration let $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} + \eta \vec{v}$, where $\eta$ is a (small) ‘step size’ and $\vec{v}$ is a direction chosen to minimize $f(\vec{\theta}^{(i-1)} + \eta \vec{v})$.

$$D_{\vec{v}} f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon \vec{v}) - f(\vec{\theta})}{\epsilon}.$$
Gradient descent is a greedy iterative optimization algorithm: Starting at \( \tilde{\theta}^{(0)} \), in each iteration let \( \tilde{\theta}^{(i)} = \tilde{\theta}^{(i-1)} + \eta \vec{v} \), where \( \eta \) is a (small) ‘step size’ and \( \vec{v} \) is a direction chosen to minimize \( f(\tilde{\theta}^{(i-1)} + \eta \vec{v}) \).

\[
D_{\vec{v}} f(\tilde{\theta}^{(i-1)}) = \lim_{\epsilon \to 0} \frac{f(\tilde{\theta}^{(i-1)} + \epsilon \vec{v}) - f(\tilde{\theta}^{(i-1)})}{\epsilon}.
\]
Gradient descent is a greedy iterative optimization algorithm: Starting at $\vec{\theta}^{(0)}$, in each iteration let $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} + \eta \vec{v}$, where $\eta$ is a (small) ‘step size’ and $\vec{v}$ is a direction chosen to minimize $f(\vec{\theta}^{(i-1)} + \eta \vec{v})$.

$$D_{\vec{v}} f(\vec{\theta}^{(i-1)}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta}^{(i-1)} + \epsilon \vec{v}) - f(\vec{\theta}^{(i-1)})}{\epsilon}.$$  

So for small $\eta$:

$$f(\vec{\theta}^{(i)}) - f(\vec{\theta}^{(i-1)}) = f(\vec{\theta}^{(i-1)} + \eta \vec{v}) - f(\vec{\theta}^{(i-1)})$$
Gradient descent is a greedy iterative optimization algorithm: Starting at $\tilde{\theta}^{(0)}$, in each iteration let $\tilde{\theta}^{(i)} = \tilde{\theta}^{(i-1)} + \eta \vec{v}$, where $\eta$ is a (small) ‘step size’ and $\vec{v}$ is a direction chosen to minimize $f(\tilde{\theta}^{(i-1)} + \eta \vec{v})$.

$$D_{\vec{v}} f(\tilde{\theta}^{(i-1)}) = \lim_{\epsilon \to 0} \frac{f(\tilde{\theta}^{(i-1)} + \epsilon \vec{v}) - f(\tilde{\theta}^{(i-1)})}{\epsilon}.$$ 

So for small $\eta$:

$$f(\tilde{\theta}^{(i)}) - f(\tilde{\theta}^{(i-1)}) = f(\tilde{\theta}^{(i-1)} + \eta \vec{v}) - f(\tilde{\theta}^{(i-1)}) \approx \eta \cdot D_{\vec{v}} f(\tilde{\theta}^{(i-1)})$$
Gradient descent is a greedy iterative optimization algorithm: Starting at \( \theta^{(0)} \), in each iteration let \( \theta^{(i)} = \theta^{(i-1)} + \eta \vec{v} \), where \( \eta \) is a (small) ‘step size’ and \( \vec{v} \) is a direction chosen to minimize \( f(\theta^{(i-1)} + \eta \vec{v}) \).

\[
D_{\vec{v}} f(\theta^{(i-1)}) = \lim_{\varepsilon \to 0} \frac{f(\theta^{(i-1)} + \varepsilon \vec{v}) - f(\theta^{(i-1)})}{\varepsilon}.
\]

So for small \( \eta \):

\[
f(\theta^{(i)}) - f(\theta^{(i-1)}) = f(\theta^{(i-1)} + \eta \vec{v}) - f(\theta^{(i-1)}) \approx \eta \cdot D_{\vec{v}} f(\theta^{(i-1)})
\]

\[
= \eta \cdot \langle \vec{v}, \nabla f(\theta^{(i-1)}) \rangle.
\]
Gradient descent is a greedy iterative optimization algorithm: Starting at $\vec{\theta}^{(0)}$, in each iteration let $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} + \eta \vec{v}$, where $\eta$ is a (small) ‘step size’ and $\vec{v}$ is a direction chosen to minimize $f(\vec{\theta}^{(i-1)} + \eta \vec{v})$.

$$D_{\vec{v}} f(\vec{\theta}^{(i-1)}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta}^{(i-1)} + \epsilon \vec{v}) - f(\vec{\theta}^{(i-1)})}{\epsilon}.$$

So for small $\eta$:

$$f(\vec{\theta}^{(i)}) - f(\vec{\theta}^{(i-1)}) = f(\vec{\theta}^{(i-1)} + \eta \vec{v}) - f(\vec{\theta}^{(i-1)}) \approx \eta \cdot D_{\vec{v}} f(\vec{\theta}^{(i-1)})$$

$$= \eta \cdot \langle \vec{v}, \nabla f(\vec{\theta}^{(i-1)}) \rangle.$$

We want to choose $\vec{v}$ minimizing $\langle \vec{v}, \nabla f(\vec{\theta}^{(i-1)}) \rangle$ – i.e., pointing in the direction of $\nabla f(\vec{\theta}^{(i-1)})$ but with the opposite sign.
Gradient Descent

• Choose some initialization $\vec{\theta}^{(0)}$.
• For $i = 1, \ldots, t$
  • $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} - \eta \nabla f(\vec{\theta}^{(i-1)})$
• Return $\vec{\theta}^{(t)}$, as an approximate minimizer of $f(\vec{\theta})$.

Step size $\eta$ is chosen ahead of time or adapted during the algorithm (details to come.)
Gradient Descent

- Choose some initialization $\vec{\theta}^{(0)}$.
- For $i = 1, \ldots, t$
  - $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} - \eta \nabla f(\vec{\theta}^{(i-1)})$
- Return $\vec{\theta}^{(t)}$, as an approximate minimizer of $f(\vec{\theta})$.

Step size $\eta$ is chosen ahead of time or adapted during the algorithm (details to come.)

- For now assume $\eta$ stays the same in each iteration.
Gradient Descent

- Choose some initialization $\vec{\theta}^{(0)}$.
- For $i = 1, \ldots, t$
  - $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} - \eta \nabla f(\vec{\theta}^{(i-1)})$
- Return $\vec{\theta}^{(t)}$, as an approximate minimizer of $f(\vec{\theta})$.

Step size $\eta$ is chosen ahead of time or adapted during the algorithm (details to come.)

- For now assume $\eta$ stays the same in each iteration.

When will this algorithm work well?
Gradient Descent Update: \( \vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} - \eta \nabla f(\vec{\theta}^{(i-1)}) \)

\( \theta \in \mathbb{R} \quad \forall f(\theta) \in \mathbb{R} \)
**Convex Functions:** After sufficient iterations, gradient descent will converge to a **approximate minimizer** \( \hat{\theta} \) with:

\[
f(\hat{\theta}) \leq f(\theta_*) + \epsilon
\]
**Convex Functions:** After sufficient iterations, gradient descent will converge to a *approximate minimizer* $\hat{\theta}$ with:

$$f(\hat{\theta}) \leq f(\theta^*) + \epsilon = \min_{\theta} f(\theta) + \epsilon.$$
Convex Functions: After sufficient iterations, gradient descent will converge to a approximate minimizer $\hat{\theta}$ with:

$$f(\hat{\theta}) \leq f(\theta^*) + \epsilon = \min_{\theta} f(\theta) + \epsilon.$$  

Examples: least squares regression, logistic regression, sparse regression (lasso), regularized regression, SVMS, ...
**Convex Functions:** After sufficient iterations, gradient descent will converge to a **approximate minimizer** $\hat{\theta}$ with:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon = \min_{\theta} f(\theta) + \epsilon.$$

Examples: least squares regression, logistic regression, sparse regression (lasso), regularized regression, SVMS,...

**Non-Convex Functions:** After sufficient iterations, gradient descent will converge to a **approximate stationary point** $\hat{\theta}$ with:

$$\|\nabla f(\hat{\theta})\|_2 \leq \epsilon.$$
**Convex Functions:** After sufficient iterations, gradient descent will converge to a **approximate minimizer** \( \hat{\theta} \) with:

\[
f(\hat{\theta}) \leq f(\theta^*) + \epsilon = \min_{\theta} f(\theta) + \epsilon.
\]

Examples: least squares regression, logistic regression, sparse regression (lasso), regularized regression, SVMS, ...

**Non-Convex Functions:** After sufficient iterations, gradient descent will converge to a **approximate stationary point** \( \hat{\theta} \) with:

\[
\|\nabla f(\hat{\theta})\|_2 \leq \epsilon.
\]

Examples: neural networks, clustering, mixture models.
Why for non-convex functions do we only guarantee convergence to a **approximate stationary point** rather than an **approximate local minimum**?
Why for non-convex functions do we only guarantee convergence to an approximate stationary point rather than an approximate local minimum?
Well-behaved functions

Gradient Descent Update: 

\[ \tilde{\theta}^{(i)} = \tilde{\theta}^{(i-1)} - \eta \nabla f(\tilde{\theta}^{(i-1)}) \]
Gradient Descent Update: $\mathbf{\theta}^{(i)} = \mathbf{\theta}^{(i-1)} - \eta \nabla f(\mathbf{\theta}^{(i-1)})$
Both Convex and Non-convex: Need to assume the function is well behaved in some way.
Both Convex and Non-convex: Need to assume the function is well behaved in some way.

- Lipschitz (size of gradient is bounded): For all $\vec{\theta}$ and some $G$,
  $$\|\vec{\nabla} f(\vec{\theta})\|_2 \leq G.$$

- Smooth (direction/size of gradient is not changing too quickly): For all $\vec{\theta}_1, \vec{\theta}_2$ and some $\beta$,
  $$\|\vec{\nabla} f(\vec{\theta}_1) - \vec{\nabla} f(\vec{\theta}_2)\|_2 \leq \beta \cdot \|\vec{\theta}_1 - \vec{\theta}_2\|_2.$$
Gradient Descent analysis for convex functions.
Definition – Convex Function: A function \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) is convex if and only if, for any \( \vec{\theta}_1, \vec{\theta}_2 \in \mathbb{R}^d \) and \( \lambda \in [0, 1] \):

\[
(1 - \lambda) \cdot f(\vec{\theta}_1) + \lambda \cdot f(\vec{\theta}_2) \geq f \left( (1 - \lambda) \cdot \vec{\theta}_1 + \lambda \cdot \vec{\theta}_2 \right)
\]
**Corollary – Convex Function:** A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if, for any $\vec{\theta}_1, \vec{\theta}_2 \in \mathbb{R}^d$ and $\lambda \in [0, 1]$:

$$f(\vec{\theta}_2) - f(\vec{\theta}_1) \geq \nabla f(\vec{\theta}_1)^T (\vec{\theta}_2 - \vec{\theta}_1)$$
Assume that:

- $f$ is convex.
- $f$ is $G$ Lipschitz (i.e., $\|\nabla f(\theta)\|_2 \leq G$ for all $\theta$).
- $\|\theta_0 - \theta_*\|_2 \leq R$ where $\theta_0$ is the initialization point.

**Gradient Descent**

- Choose some initialization $\theta_0$ and set $\eta = \frac{R}{G\sqrt{t}}$.
- For $i = 1, \ldots, t$
  - $\theta_i = \theta_{i-1} - \eta \nabla f(\theta_{i-1})$
- Return $\hat{\theta} = \arg\min_{\theta_0, \ldots, \theta_t} f(\theta_i)$. 
Theorem – GD on Convex Lipschitz Functions: For convex $G$ Lipschitz function $f$, GD run with $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G \sqrt{t}}$, and starting point within radius $R$ of $\theta_*$, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$
Theorem – GD on Convex Lipschitz Functions: For convex $G$ Lipschitz function $f$, GD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius $R$ of $\theta_*$, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$ 

Step 1: For all $i$, $f(\theta_i) - f(\theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$. Visually:
Theorem – GD on Convex Lipschitz Functions: For convex $G$ Lipschitz function $f$, GD run with $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius $R$ of $\theta_\star$, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_\star) + \epsilon.$$ 

Step 1: For all $i$, $f(\theta_i) - f(\theta_\star) \leq \frac{||\theta_i - \theta_\star||_2^2 - ||\theta_{i+1} - \theta_\star||_2^2}{2\eta} + \frac{\eta G^2}{2}$. Formally:
Theorem – GD on Convex Lipschitz Functions: For convex $G$ Lipschitz function $f$, GD run with $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G \sqrt{t}}$, and starting point within radius $R$ of $\theta_*$, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

Step 1: For all $i$, $f(\theta_i) - f(\theta_*) \leq \frac{||\theta_i - \theta_*||^2_2 - ||\theta_{i+1} - \theta_*||^2_2}{2\eta} + \frac{\eta G^2}{2}$.

Step 1.1: $\nabla f(\theta_i)(\theta_i - \theta_*) \leq \frac{||\theta_i - \theta_*||^2_2 - ||\theta_{i+1} - \theta_*||^2_2}{2\eta} + \frac{\eta G^2}{2}$.
Theorem – GD on Convex Lipschitz Functions: For convex $G$ Lipschitz function $f$, GD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius $R$ of $\theta_*$, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

Step 1: For all $i$, $f(\theta_i) - f(\theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$.

Step 1.1: $\nabla f(\theta_i)(\theta_i - \theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \implies$ Step 1.
Theorem – GD on Convex Lipschitz Functions: For convex $G$ Lipschitz function $f$, GD run with $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius $R$ of $\theta_*$, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$ 

Step 1: For all $i$, $f(\theta_i) - f(\theta_*) \leq \frac{\|\theta_i - \theta_*\|^2_2 - \|\theta_{i+1} - \theta_*\|^2_2}{2\eta} + \frac{\eta G^2}{2}$.
Theorem – GD on Convex Lipschitz Functions: For convex $G$ Lipschitz function $f$, GD run with $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G \sqrt{t}}$, and starting point within radius $R$ of $\theta_*$, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$ 

Step 1: For all $i$, $f(\theta_i) - f(\theta_*) \leq \frac{||\theta_i - \theta_*||^2}{2\eta} - \frac{||\theta_{i+1} - \theta_*||^2}{2\eta} + \frac{\eta G^2}{2} \implies$

Step 2: $\frac{1}{T} \sum_{i=1}^{T} f(\theta_i) - f(\theta_*) \leq \frac{R^2}{2\eta \cdot T} + \frac{\eta G^2}{2}$.
Theorem – GD on Convex Lipschitz Functions: For convex $G$ Lipschitz function $f$, GD run with $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G \sqrt{t}}$, and starting point within radius $R$ of $\theta_*$, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

Step 2: $\frac{1}{T} \sum_{i=1}^{T} f(\theta_i) - f(\theta_*) \leq \frac{R^2}{2\eta \cdot T} + \frac{\eta G^2}{2}$. 
Questions on Gradient Descent?